Freescale MQX™ RTOS 3.1 Release Notes

PRODUCT:	Freescale MQX™ RTOS
PRODUCT VERSION:	3.1.0
DESCRIPTION:	Freescale MQX™ RTOS Operating System, version 3.1
RELEASE DATE:	April 3, 2009

© Freescale Semiconductor, Inc., 2009. All rights reserved.

freescale™ semiconductor

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or +1-303-675-2140
Fax: +1-303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typicals”, must be validated for each customer application by customer’s technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. ARC, the ARC logo, ARCangel, ARCform, ARChitect, ARCompact, ARCtangent, BlueForm, CASSEIA, High C/C++, High C++, iCon186, MetaDeveloper, MQX, Precise Solution, Precise/BlazeNet, Precise/EDS, Precise/MFS, Precise/MQX, Precise/MQX Test Suites, Precise/RTCS, RTCS, SeeCode, TotalCore, Turbo186, Turbo86, V8 µ RISC, V8 microRISC, and VAutomation are trademarks of ARC International. High C and MetaWare are registered under ARC International.

All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc. 2009. All rights reserved.

Rev. 1
3/2009

Table of Contents

Freescale MQX™ RTOS 3.1 Release Notes i

1 Read Me First 2
1.1 Requirements 2
1.2 Special instructions 2

2 Release Content 3

3 Last Version Highlights 6
3.1 MQX RTOS PSP 6
3.2 MQX RTOS BSPs 6
3.3 MQX MFS 9
3.4 MQX RTCS 10
3.5 MQX USB Host 10
3.6 MQX USB Device 11
3.7 MQX Shell 11
3.8 Compile-time Configuration 11
3.9 Changing the MQX Source Files 12
3.10 Building the MQX Libraries 12
3.11 Example Applications 14
CodeWarrior Task Aware Debugging Plug-in 18

4 What is Missing? 21
4.1 In MQX PSP 21
4.2 In RTCS TCP/IP Stack 21
4.3 In USB Support 22
4.4 Peripheral Drivers 22
4.5 PC Host Tools 22

5 Known Issues 23
5.1 Performance of Code Running in MRAM 23
5.2 Network Communication Performance and Delayed ACKs 23
5.3 “HVAC” Demo Applications 23
5.4 Ethernet Link-status Detection 24
5.5 Default Kernel Configuration 24
5.6 USB Host HUB Examples 25
5.7 USB Host HUB Functionality with MCF5222x 25
5.8 BDM Connection Failures with MCF5223x 25
5.9 MQX Stationery / CodeWarrior FlashProgrammer Issue 25

6 Change Log 28

1 Read Me First

This release note documents the Freescale MQX™ RTOS version 3.1 released for Freescale ColdFire® processor family.

1.1 Requirements

1.1.1 System Requirements

This Freescale MQX™ RTOS Release was compiled and tested with CodeWarrior for ColdFire 7.1 (patch version 7.1.1a, Build 9), IDE version 5.9.0. The system requirements are defined by the CodeWarrior requirements. There is no special host system requirement for hosting the Freescale MQX™ RTOS distribution itself.

Minimum PC configuration:

- As required by CodeWarrior for ColdFire 7.1

Recommended PC configuration:

- 2 GHz processor – 1 GB RAM - 1 GB free disk space

Software requirements:

- OS: As required by CodeWarrior for ColdFire 7.1 (Windows XP SP2 or later recommended)

1.1.2 Target Requirements

The Freescale MQX™ RTOS in this release supports the evaluation boards mentioned below. There are no special requirements for the target hardware which would be out of scope of what each board requires for its operation (power supply, cabling, etc).

- M52259EVB Evaluation Board
- M52259DEMOKIT Evaluation Board
- M52235EVB Evaluation Board
- M52223EVB Evaluation Board

1.2 Special instructions

1.2.1 Setup Installation instructions

Run the self-extracting installer application and proceed according to instructions.

In case you change the default installation location (which is C:\Program Files\Freescale\Freescale MQX 3.1), it is recommended to re-compile all core libraries. Otherwise, any time the application is started under debugger, the debugger may ask for a path to MQX source code files.

To re-build the libraries, open and build the following CodeWarrior project:

```
<install_dir>/config/<board>/build_libs.mcp
```

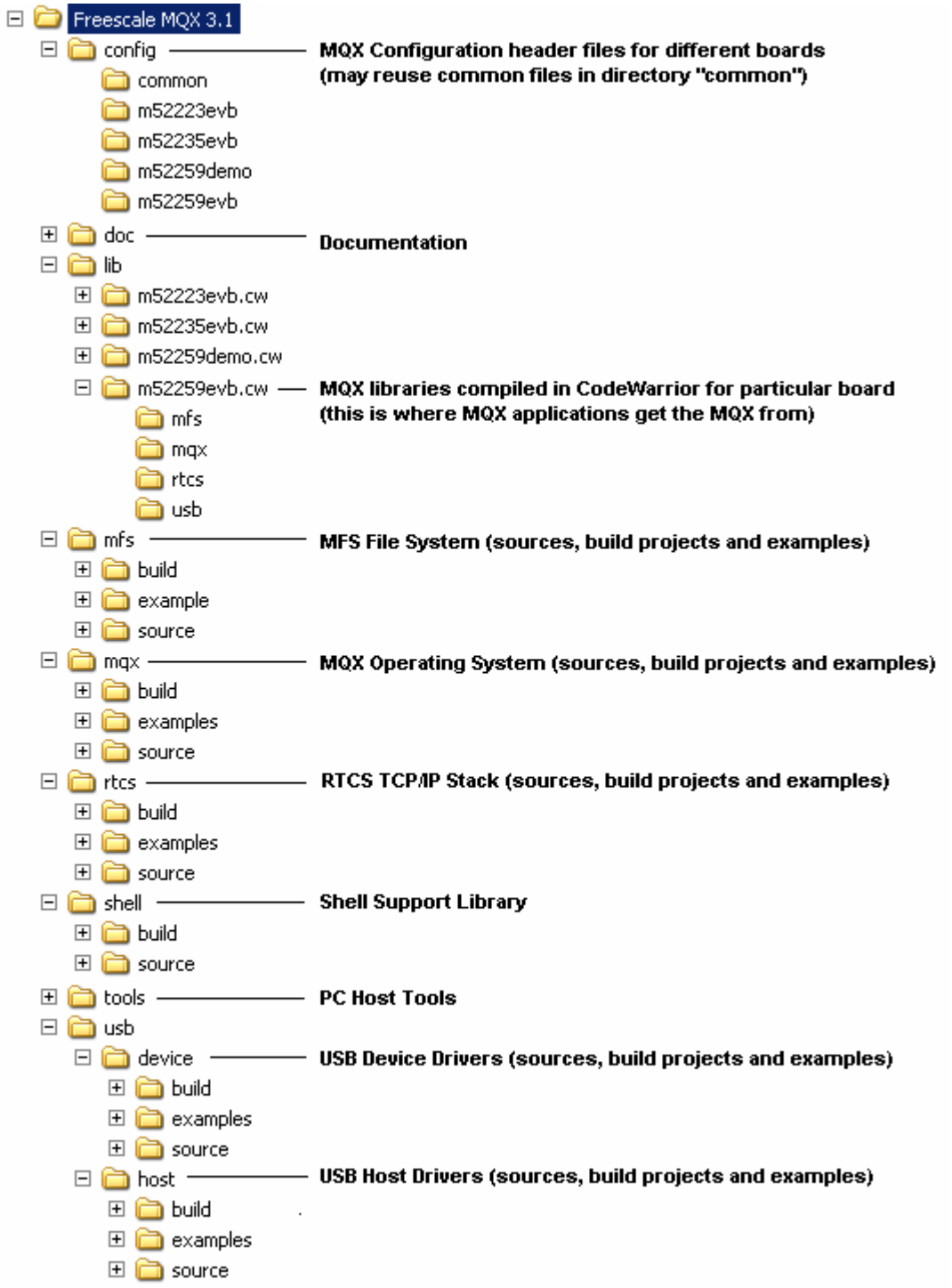
See more details about re-building the MQX libraries in the following sections.

2 Release Content

Deliverable	Location	Status
Pre-compiled MQX Libraries	**<install_dir>/lib/...**	
MQX PSP and BSP	.../lib/<board>/mqx	updated
MQX RTCS (TCP/IP network stack)	.../lib/<board>/rtcs	updated
MQX MFS (File System)	.../lib/<board>/mfs	updated
MQX USB Host Drivers	.../lib/<board>/usb/host	new location
MQX USB Device Drivers	.../lib/<board>/usb/device	new
MQX Shell Library	.../lib<board>/shell	updated
Configuration Files and Mass-Build Projects	**<install_dir>/config/...**	
Configuration and mass-build project for M52259EVB board	.../config/m52259evb	updated
Configuration and mass-build project for M52259DEMO board	.../config/m52259demo	updated
Configuration and mass-build project for M52235EVB board	.../config/m52235evb	new
Configuration and mass-build project for M52223EVB board	.../config/m52223evb	new
MQX PSP Source Code and Examples	**<install_dir>/mqx/...**	
MQX PSP source code for ColdFire	.../mqx/source/psp/coldfire	updated
MQX PSP build projects	.../mqx/build/codewarrior/psp_*	updated
MQX example applications	.../mqx/examples/...	updated
MQX BSP Source Code	**<install_dir>/mqx/...**	
MQX BSP source code for M52259EVB board	.../mqx/source/bsp/m52259evb	updated
MQX BSP source code for M52259DEMO board	.../mqx/source/bsp/m52259demo	updated
MQX BSP source code for M52235EVB board	.../mqx/source/bsp/m52235evb	new
MQX BSP source code for M52223EVB board	.../mqx/source/bsp/m52223evb	new
MQX BSP build projects	.../mqx/build/codewarrior/bsp_*	updated
RTCS Source Code and Examples	**<install_dir>/rtcs/...**	
RTCS source code	.../rtcs/source	updated
RTCS build projects	.../rtcs/build/codewarrior	updated
RTCS example applications	.../rtcs/examples	updated
MFS Source Code and Examples	**<install_dir>/mfs/...**	
MFS source code	.../mfs/source	updated
MFS build projects	.../mfs/build/codewarrior	updated
MFS example applications	.../mfs/examples	updated
USB Host Drivers Source Code and Examples	**<install_dir>/usb/host/...**	
USB Host source code and class drivers	.../usb/host/source	updated
HUB Class Driver	.../usb/host/source/hub	updated
Human Interface Device (HID) Class Driver	.../usb/host/source/hid	updated
Mass Storage (MASS) Class Driver	.../usb/host/source/mass	updated
Printer Class Driver	.../usb/host/source/printer	from 3.0
USB Host build projects	.../usb/host/build/codewarrior	updated
USB Host example applications (HID, MASS)	.../usb/host/examples	updated
USB Device Drivers Source Code and Examples	**<install_dir>/usb/device/...**	
USB Device source code	.../usb/device/source	new
USB Device build projects	.../usb/device/build/codewarrior	new
USB Device example applications (HID, MASS)	.../usb/device/examples	new

Shell Library Source Code	**<install_dir>/shell/...**	
Shell source code	.../shell/source	updated
Shell build projects	.../shell/build/codewarrior	updated
CodeWarrior Support	**<CodeWarrior_dir>/...**	
MQX New project stationery	<cw_dir>/stationery/Freescale MQX 3.1	new
MQX Task-aware Debugger plug-in (TAD)	<cw_dir>/bin/plugins/debugger/rtos/	updated
PC Host Tools	**<install_dir>/tools**	
TFS Make Utility	…/tools/mktfs.exe	from 3.0
AWK interpreter (GNU General Public License)	…/tools/gawk.exe	new
SNMP code generation scripts	…/tools/snmp/*.awk	new
Demo Applications	**<install_dir>/demo**	
Various demo applications described in detail by step-by-step ‘Lab’ documents.	…/demo/...	updated

The following picture shows the Freescale MQX™ RTOS directories installed to the user host computer:



3 Last Version Highlights

This is the MQX RTOS release done by Freescale Semiconductor, targeting the MCF522xx ColdFire family of microcontrollers. It is based on the version 2.50 as it was released by the company ARC International. Freescale MQX™ RTOS contains also other core system components like RTCS network stack, MFS file system and USB host drivers which used to be released and sold separately by ARC.

In addition to changed directory structure, the Freescale release also brings a new way of how user configures the kernel components before compiling. See more details in sections below.

3.1 MQX RTOS PSP

This version of Freescale MQX™ RTOS contains ColdFire Platform Support Package only. Support for PowerPC platform and other Freescale processor platforms is planned for future releases.

The major PSP modifications as compared to the ARC release:

- The compile-time configuration of the MQX PSP (as well as all the other MQX components) is now accomplished by editing the `.../config/<board>/user_config.h` file. In the previous versions, the user configuration macros were passed to the build process on a command line by the makefile or CodeWarrior prefix file.
- The PSP is configured and built for specific board. Originally, the PSP component was built once for the particular device. While the PSP code still remains board-independent, this feature enables finer kernel tuning for a specific board.
- MQX now supports typed memory, which allows additional information to be displayed in the Task Aware Debugger Plugin.

3.2 MQX RTOS BSPs

This release includes Board Support Package for boards mentioned below. In addition to the board initialization code, BSPs contain I/O drivers suitable for each of the boards. The following list gives an overview about supported drivers:

All boards:

- **TFS – Trivial Filesystem** – tiny filesystem which can be used as a simple read-only file repository instead of fully featured MFS. TFS is not installed in BSP startup code. It is a task for the application to initialize the TFS and pass a pointer to the filesystem image data. The **mktfs** tool is available (both as executable and PERL script) to generate the image from the existing directory structure. The RTCS HTTP example demonstrates use of TFS.
- **I2C I/O Driver** – This driver supports a polled I2C interface in master mode. If enabled in user configuration the I2C device drivers are installed during the BSP startup code as "**i2c0:**" and "**i2c1:**". Example application is provided in the MQX source tree.
- **QSPI I/O Driver** – This driver supports a polled QSPI interface in master mode. If enabled in user configuration the QSPI device drivers is installed during the BSP startup code as "**qspi:**". Example application is provided in the MQX source tree.
- **FlexCAN Driver** – This driver provides a C language API to the FlexCAN peripheral module. It is not a true I/O driver as defined by MQX terminology. Example application is provided in the MQX source tree.

- **RTC Driver** – This driver provides a C language API to the Real Time Clock peripheral module and functions helping to synchronize clock time between RTC and MQX system. If enabled in user configuration, the RTC module is initialized and MQX time is renewed automatically during BSP startup.
- **Serial Driver** – The standard UART driver supporting both polled and interrupt-driven modes. If enabled in user configuration, the serial devices are installed as "**ttya:**", "**ttyb:**" and "**ttyc:**" (polled mode) and "**ittya:**", "**ittyb:**" and "**ittyc:**" (interrupt mode) automatically during BSP startup.
- **GPIO Driver** – The I/O driver providing a uniform interface to all GPIO pins available. If enabled in user configuration, the GPIO driver is installed as "**gpio:**" automatically by the BSP startup.
- **ADC Driver** – The I/O driver providing a uniform interface to ADC channels. ADC uses internal PIT timer for periodic sampling. If enabled in user configuration, the ADC driver is installed as "**adc:**" automatically by the BSP startup.

  This release contains an initial version of the driver. The code (including the API) is subject to improvement in future releases.

  Current version limitations (features planned for the next version):

  - The *close* operation is not implemented. ADC driver and virtual channel "files" can be initialized and opened with `fopen`, but not closed.
  - Channel resolution setting & scaling is not yet implemented.
  - "Data available" Event-based notification is not yet implemented.
  - Once a channel sampling is started, it stops after the sequence is finished (ONCE) or runs continuously (LOOP). Stop/abort sampling is not yet implemented.
  - The driver API is not yet documented. There is a simple example application demonstrating the basic driver features.

Ethernet-based Boards (M52259x and M5223x)

- **Ethernet Driver** – The low-level Ethernet driver used by the RTCS TCP/IP software stack. The driver is initialized directly by the application before RTCS is first used. The RTCS HTTP example demonstrates use of this driver.

M52259EVB Board Only

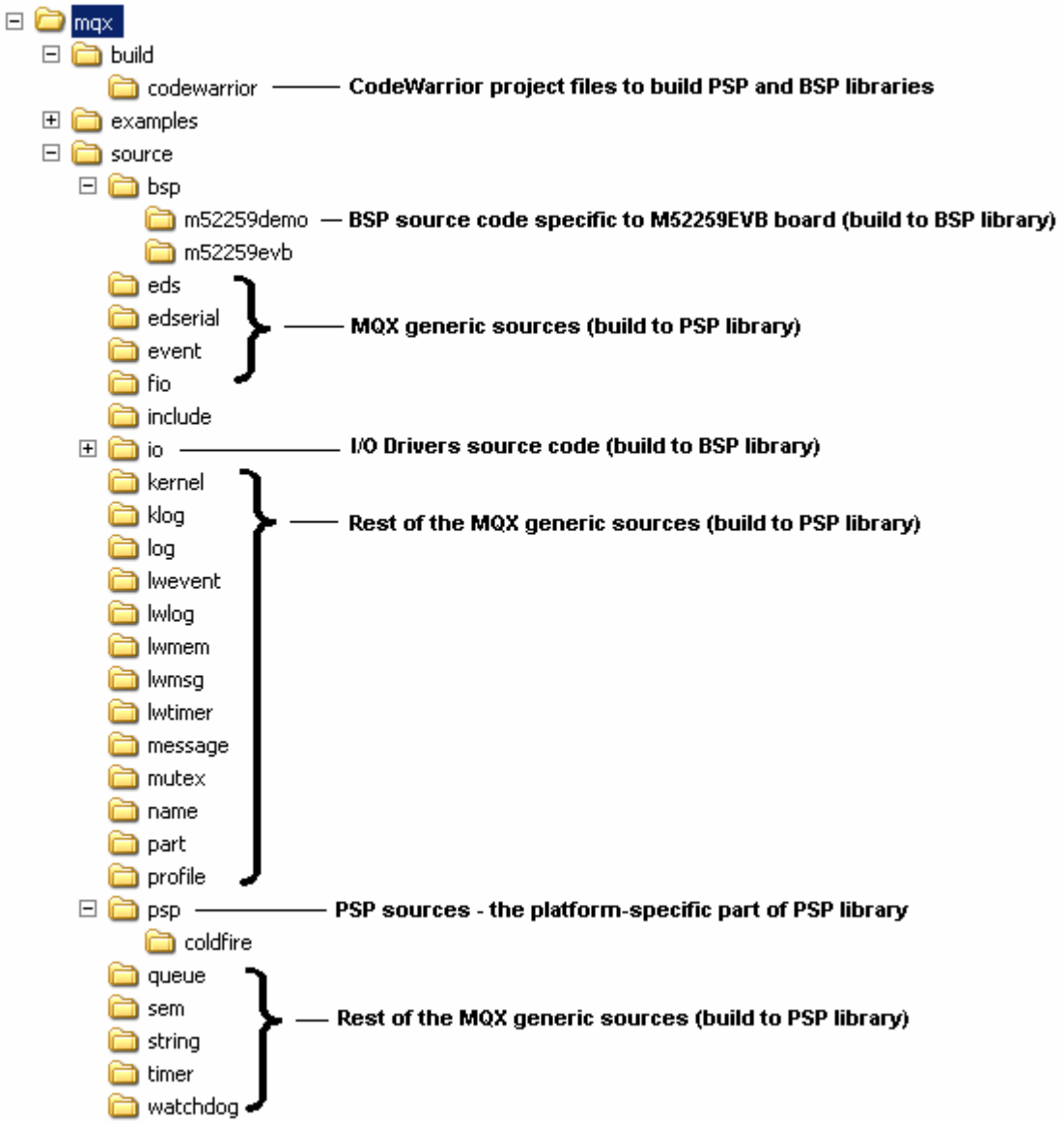
- **PCCard Driver** – The I/O driver providing low-level access to the PCCard functionality implemented by CPLD on M52259EVB board. If enabled in user configuration, the PCCard device driver is installed as "**pccarda:**" automatically during the BSP startup.
- **PCFlash Driver** – The Compact Flash card I/O driver which uses the PCCard low-level driver and enables standard drive operations. The MFS file system can be installed on top of this device. If enabled in user configuration, the PCFlash device driver is installed as "**pcflasha:**" automatically during the BSP startup.

The following table gives an overview about I/O drivers which can be installed automatically by the BSP startup code. To save Code memory and RAM, most of the drivers are disabled in the `user_config.h` file by default.

I/O Device	Peripheral Module	Configuration Macro (user_config.h)	Enabled by default			
			M52259EVB	M52259DEMO	M52235EVB	M52223EVB
ttya:	UART0	BSPCFG_ENABLE_TTYA	yes	yes	yes	yes
ttyb:	UART1	BSPCFG_ENABLE_TTYB	no	no	no	no
ttyc:	UART2	BSPCFG_ENABLE_TTYC	no	no	no	no
ittya:	UART0	BSPCFG_ENABLE_ITTYA	no	no	no	no
ittyb:	UART1	BSPCFG_ENABLE_ITTYB	no	no	no	no
ittyc:	UART2	BSPCFG_ENABLE_ITTYC	no	no	no	no
i2c0:	I2C0	BSPCFG_ENABLE_I2C0	no	no	no	no
i2c1:	I2C1	BSPCFG_ENABLE_I2C1	no	no	no	no
qspi:	QSPI	BSPCFG_ENABLE_QSPI	no	no	no	no
gpio:	GPIO	BSPCFG_ENABLE_GPIODEV	yes	yes	yes	yes
adc:	ADC	BSPCFG_ENABLE_ADC	no	no	no	no
-	RTC	BSPCFG_ENABLE_RTCDEV	no	no	no	no
pccarda: pcflasha:	FlexBus + CPLD	BSPCFG_ENABLE_PCFLASH	yes	n/a	n/a	n/a

3.2.1 MQX RTOS Directory Structure

The RTOS files are located in the `mqx` subdirectory of the Freescale MQX™ RTOS installation. The directory structure is briefly described in the picture below.



3.3 MQX MFS

This release Freescale MQX™ RTOS includes the MFS File System libraries. The MFS is based on the version 2.20 released by ARC.

The major MFS modifications as compared to the ARC release:

- The compile-time configuration of the MFS is now accomplished by editing the `.../config/<board>/user_config.h` file.
- MFS was significantly optimized for devices with limited RAM resources. In particular, MFS now performs sector caching, not cluster caching. This generally results in better performance with a lower RAM footprint.

- MFS supports allocation of resources from a specific memory pool.
- Compile time and run-time configurable options were added in order to selectively enable and disable individual MFS features.
- Similarly as the PSP, the MFS is configured and built for specific board. Originally, the MFS component was built once for the particular device or even the whole ColdFire family. While the MFS code still remains board-independent, this feature enables finer feature tuning for a specific board.

3.4 MQX RTCS

This release of Freescale MQX includes the RTCS TCP/IP stack libraries. The RTCS is based and is API-level compatible with version 2.97 released by ARC.

The major RTCS modifications as compared to the ARC release:

- The compile-time configuration of the RTCS is now accomplished by editing the `.../config/<board>/user_config.h` file.
- Similarly as the PSP, the RTCS is configured and built for specific board. Originally, the RTCS component was built once for the particular device or even the whole ColdFire family. While the RTCS code still remains board-independent, this feature enables finer feature tuning for a specific board.
- The RTCS was significantly optimized for devices with limited RAM resources. Several configurable options were added in order to selectively enable and disable individual TCP/IP features and protocols.
- The new FTP Client and Server have been moved from the example directory to the apps directory.
- IPCFG feature in RTCS allows to monitor Ethernet link status and performs automated IP address binding, either static IP, DHCP or DHCP with Auto IP assignment. Status may be monitored in dedicated background task or by polling calls. New “ipconfig” shell command is available as a user interface to the IPCFG functionality.
- RTCS supports allocation of resources from a specific memory pool.
- The HTTP server code was added. This server enables standard to implement web server with a support for CGI-like callbacks or ASP-like page placeholder callbacks.

3.5 MQX USB Host

This release of Freescale MQX includes the USB Host drivers and USB class drivers. The USB code is based on version 1.2.0 released by ARC.

The major USB modifications as compared to the ARC release:

- The compile-time configuration of the USB Host library is now accomplished by editing the `.../config/<board>/user_config.h` file.
- A new USB Host low-level driver was written for MCF5222x and MCF5225x devices. This new driver does not imply any change to higher levels of the USB Host stack. The driver codename is KHCI.
- The separate Device-specific Driver, Host Development Kit library and USB Class library projects were merged into a single library called HDK. The board-specific code was moved to BSP library.
- Similarly as the PSP, the HDK library is now configured and built for specific board.

List of USB class drivers included in the Host Development Kit

- Human-interface Class (HID) used to access mouse, keyboard and similar devices.
- Mass storage Class used to access USB drives.
- USB Hub class used to attach multiple devices to a single host port. If enabled at the application level, the HUB support is fully transparent. User application only needs to be modified to be ready to handle multiple USB devices simultaneously. Keyboard/Mouse example application is provided.
- Basic Printer class.

3.6 MQX USB Device

This release of Freescale MQX includes the USB Device drivers and example applications.

Features:

- The compile-time configuration of the USB Device Development Kit (DDK) library is now accomplished by editing the `.../config/<board>/user_config.h` file.
- A new USB Device low-level driver was written for MCF5222x and MCF5225x devices. This new driver does not imply any change to higher levels of the USB Device applications.
- Similarly as the PSP, the DDK library is now configured and built for specific board.
- The USB Device communication stack implements a generic API and does not contain any specific class drivers. The class support is implemented at the application level. This concept is subject to change in future versions.
- The following standard class example applications are provided:
    - HID (mouse emulation)
    - Mass Storage (internal RAM area accessed as mass storage)

3.7 MQX Shell

The shell and command-line handling code was moved from RTCS into a separate library called Shell. Similarly as other components, the shell is configured and built for specific board.

3.8 Compile-time Configuration

One of the major changes when comparing the Freescale release with the original ARC release is the approach to compile-time configuration of kernel and other core components. All compile-time configuration options were centralized into a single user configuration file located in

```
<install_dir>/config/<board>/user_config.h
```

This user configuration file is included internally by private configuration files in MQX PSP and BSP.

To share configuration settings between different boards, the user_config.h may include other header files with common settings. The header files may only be located in the same <board> directory or in the common directory:

```
<install_dir>/config/common
```

All MQX configuration files are also *indirectly* used by other core components like RTCS, MFS, etc. “Indirectly” means that the MQX PSP and BSP must be build first, which causes the configuration file being copied into the output (lib) directory. The other components then include the configuration file from the lib output directory.

Until the PSP or BSP libraries are rebuilt, the configuration change made in the user_config.h file is not used by any other MQX component. On the other hand, after the PSP and BSP libraries are re-compiled with a new configuration, it is important to recompile the other libraries so the compiled code is consistent with the configuration file. See the next section for more details.

3.9 Changing the MQX Source Files

Although the Freescale MQX RTOS is distributed in the source code form, it is strongly recommended NOT to modify any of the source files other than compile-time configuration files.

This recommendation applies to all files under “source” and “build” sub-directories in all MQX, RTCS, MFS, USB and other core components folders.

Only in case of creating custom board support packages or adding additional I/O drivers, there are two directories where the new files and subdirectories may need to be added:

```
<install_dir>/mqx/source/bsp
<install_dir>/mqx/source/io
```

3.10 Building the MQX Libraries

3.10.1 Build Process

After any change to the compile-time user configuration file or MQX kernel source files, the MQX libraries need to be re-built. The build process is similar with all core components:

- The output directory is `<install_dir>/lib/<board>.cw/<component>`

  For example the MQX PSP and BSP libraries for M52259EVB board are built in the `.../lib/m52259evb.cw/mqx` directory.
- All public header files needed by application to make use of the library are copied from internal include folders to the same output directory as the library itself.
- During PSP or BSP build process, also the user_config.h file and other header files from the `config/<board>` and `config/common` directories are copied into the `.../lib/<board>.cw` output directory.
- Other components like RTCS, MFS, etc. use the copied configuration files only.
- Similarly as the PSP and BSP, the other libraries build into the output directories inside `.../lib/<board>.cw`

To summarize the points above, there are simple rules to obey when re-building the MQX libraries.

- After any change to the `.../config/common/user_config.h` file, all MQX libraries should be re-built.
- The PSP and BSP libraries must be build first, before the RTCS, MFS and other libraries.
- No changes should be made to header files in the output build directory. The files get overwritten any time the libraries are built.

To build all MQX libraries at once, the special mass-build project may be used. The project is located together with the user_config.h file in directory:

```
<install_dir>/config/<board>/build_libs.mcp
```

3.10.2 Debug and Release Builds

Each CodeWarrior project in Freescale MQX™ RTOS contains multiple compiler/linker configurations (so called build „targets“). There are always three build targets in any MQX core component:

- **Debug** – the compiler optimizations are turned off or set to low. The compiled code is easy to debug but may be less effective than the Release build. All output libraries (or executables) have **_d** postfix in the file name (e.g. `rtcs_d.lib`).
- **Release** – the compiler optimizations are set to maximum. The compiled code is very hard to debug and should be used for final applications only. There is no postfix in the output file name (e.g. `rtcs.lib`).
- **Build all** –this target contains all other targets for convenient mass-build.

The Debug/Release build targets exist also in all application projects. As application projects are targeted always to specific board, there are also other build targets available:

- **Int. Flash Release** – this target is suitable for final application deployment. When programmed to Flash, the application starts immediately after reset. Variables are allocated in internal SRAM memory. Application executable need to be written to Flash memory using the Flash Programmer tool.
- **Int. Flash Debug** – same as above, only the Debug-compiled libraries are used. This target is suitable for debugging before deployment. On boards without external memory, this is the only target suitable for debugging larger applications. Application executable need to be written to Flash memory using the Flash Programmer tool.
- **Int. SRAM Debug** – solely for debugging purposes with both code and variables allocated in the internal SRAM memory. This target is unusable with larger applications. Application executable is loaded to RAM automatically by the debugger.
  Note: in this target, the RAM area of is split to Code and Data parts in approximate ratio of 8:1. It may happen that this split does not suit well all applications and may need to be changed in linker command file (`intram.lcf`). Remember that this target is only suitable for tiny MQX-only applications. This target is not available in applications where the application can not fit into memory (RTCS, MFS or USB applications).
- **Ext. MRAM Debug** – solely for debugging purposes with code located in external MRAM memory (available e.g. on the M52259EVB). Variables are located in internal SRAM. Part of the external MRAM memory may also be used for additional RAM memory pools. Application executable is loaded to MRAM automatically by the debugger.

The following table gives an overview about what build targets are available for which BSP/board.

Target Name	Availability				Comment
	M52259EVB	M52259DEMO	M52235EVB	M52223EVB	
Int. Flash Release	●	●	●	●	
Int. Flash Debug	●	●	●	●	
Int. SRAM Debug		●			Only selected M52259DEMO applications use this target.
Ext. MRAM Debug	●				

3.11 Example Applications

There are example „Lab“ applications in the directory:

```
<install_dir>/demo
```

The examples are accompanied with Lab guide documents – describing step-by-step how to run them on the target board. The examples were written to demonstrate the most frequently used features of the Freescale MQX™ RTOS.

In addition to these demo applications, there are simpler example applications available in MQX, RTCS, MFS and USB Host directories.

The following tables summarize example applications provided with each system component.

MQX Example Applications

`mqx/examples/...`

Name	Availability				Description
	M52259EVB	M52259DEMO	M52235EVB	M52223EVB	
adc	●	●	●	●	Shows usage of the ADC driver, sampling analog values from two ADC channels.
can/flexcan	●	●	●		Shows usage of FlexCAN API functions to transmit and receive standard or remote frames.
demo	●	●	●	●	Shows MQX multitasking and inter-process communication using standard objects like semaphores, events or messages. See lwdemo for the same example using the lightweight objects.
event	●	●	●	●	Simple demonstration of MQX events.
gpio	●	●	●	●	Shows usage of GPIO driver to control on-board LEDs and switches.
hello	●	●	●	●	A trivial Hello World application using a single task.
hello2	●	●	●	●	A trivial Hello World application spread across two tasks.
i2c	●	●	●	●	Shows how to read/write data from/to external SPI EEPROM. Additional HW setup is needed.
io	●	●	●	●	Shows using an alternate UART port as a console output.

isr	●	●	●	●	Shows how to install interrupt service routine and how to chain it with the previous handler.
klog	●	●	●	●	Shows kernel events being logged and later the log entries dumped on console.
log	●	●	●	●	Shows application-specific logging feature.
lwdemo	●	●	●	●	Same as the "demo" application, but implemented using lightweight components only.
lwevent	●	●	●	●	Simple demonstration of MQX lightweight events.
lwlog	●	●	●	●	Simple demonstration of MQX lightweight log feature.
lwmsgq	●	●	●	●	Simple demonstration of MQX lightweight inter-process messaging.
lwsem	●	●	●	●	Simple demonstration of MQX task synchronization using the lightweight semaphore object.
msg	●	●	●	●	Simple demonstration of MQX inter-process message passing.
mutex	●	●	●	●	Simple demonstration of MQX task synchronization using the mutex object.
nill	●	●	●	●	Yet simpler than Hello World. A void application which may be used for copy/paste to start custom application.
qspi	●		●	●	Shows how to read/write data from/to external SPI EEPROM. Additional HW setup is needed.
rtc	●	●	●	●	Shows the Real Time Clock module API. Demonstrates how to synchronize RTC and MQX time and how to use RTC alarm interrupts.
sem	●	●	●	●	Simple demonstration of MQX task synchronization using the semaphore object.
taskat	●	●	●	●	Shows how task can be created within statically allocated memory buffer (avoid heap allocation for task stack and context).
taskq	●	●	●	●	Shows custom task queue and how the queue can be suspended and resumed.
test	●	●	●	●	Shows the self-testing feature of each MQX component.
tfs	●	●	●	●	Shows the usage of ROM-based Trivial File System in an MQX application.
timer	●	●	●	●	Simple demonstration of MQX timer component.
watchdog	●	●	●	●	Simple demonstration of MQX task timeout detection using the watchdog component.

RTCS Example Applications

`rtcs/examples/...`

Name	Availability				Description
	M52259EVB	M52259DEMO	M52235EVB	M52223EVB	
httpsrv	●	●	●		Simple web server with cgi scripts and few web pages stored in internal flash.
shell	●	●	●		Shell command line providing commands for network management.
snmp	●	●	●		SNMP protocol example providing microprocessor state information.

MFS Example Applications
`mfs/examples/...`

Name	Availability				Description
	M52259EVB	M52259DEMO	M52235EVB	M52223EVB	
`mfs_ftp`	●	●			RTCS FTP demo accessing MFS filesystem mounted on the USB mass storage.
`mfs_usb`	●	●		●	Console shell-based example showing how to access MFS filesystem mounted on the USB mass storage.
`mfs_cfcard_usb`	●				Console shell-based example showing the MFS filesystem used with both USB and CFCard storage. Files can be copied between the two devices.

USB Host Example Applications
`usb/host/examples/...`

Name	Availability				Description
	M52259EVB	M52259DEMO	M52235EVB	M52223EVB	
`hid/keyboard`	●	●		●	This application echoes keys pressed on the USB keyboard onto serial console.
`hid/mouse`	●	●		●	Displays USB mouse events on serial console.
`hid/keyboard+mouse`	●	●		●	Keyboard and mouse demos combined in a single application.
`mass/msc_commands`	●	●		●	Executes the standard "mass storage device" commands to the USB disk and shows the response on the serial console (see mfs demos for USB filesystem access).

USB Device Example Applications
`usb/device/examples/...`

Name	Availability				Description
	M52259EVB	M52259DEMO	M52235EVB	M52223EVB	
`disk`	●	●			Implements small storage device in internal RAM memory.
`mouse`	●	●		●	Creates virtual mouse which keeps moving in square loop, 100 pixels in size.

Lab Tutorial Demos

`demo/...`

Name	Availability				Description
	M52259EVB	M52259DEMO	M52235EVB	M52223EVB	
`hvac`	●	●			Simple implementation of console-based HVAC, with optional USB logging and FTP access.
`hvac_error`	●	●			Intentional error injected to the HVAC demo code to demonstrate power of TAD plug-in.
`web_hvac`	●	●			The HVAC demo with HTTP server implementing the GUI. Ajax-based pages demonstrate an advanced use of HTTP server.
`telnet_to_serial`	●	●	●		Simple character passing between UART console and telnet session. Shows custom "lightweight" telnet server.

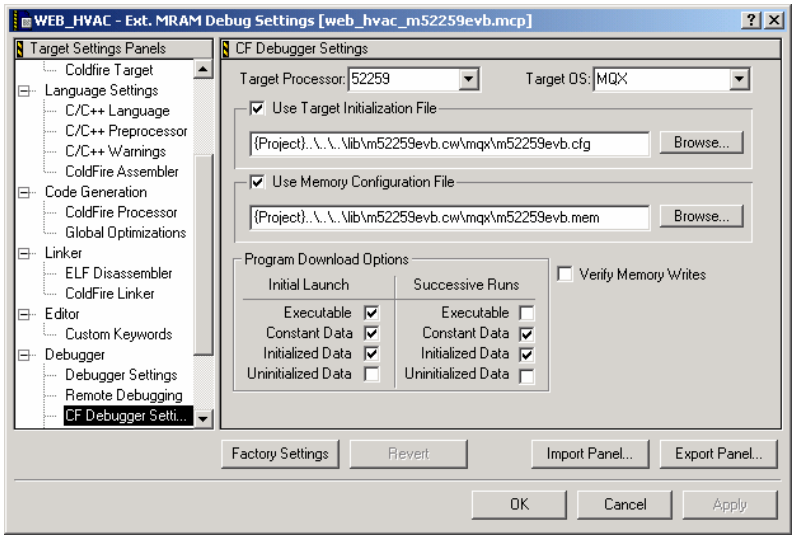
CodeWarrior Task Aware Debugging Plug-in

Freescale MQX™ RTOS brings a new version of Task Aware Debugging Plug-in (so called TAD) for CodeWarrior Development Studio. Comparing it to the previous ARC release, the TAD was significantly re-written in order to achieve better stability and portability across different CodeWarrior versions.

3.11.1 Installing CodeWarrior TAD

TAD plug-in DLL is installed into the selected CodeWarrior tool automatically during Freescale MQX™ RTOS setup process. Perform the following steps to install TAD manually (for example to a newly installed CodeWarrior studio):

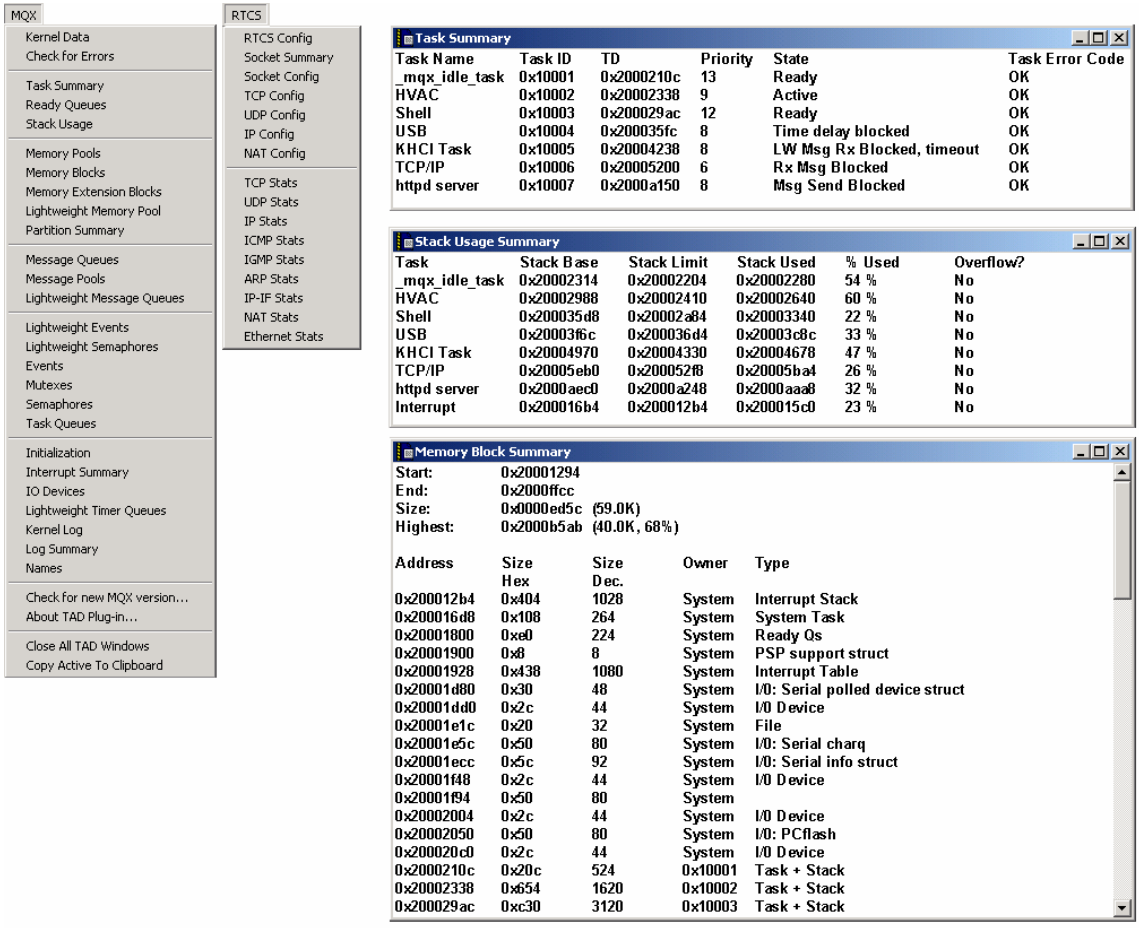
1. Locate the *tools* directory in Freescale MQX™ RTOS installation folder (by default `C:\Program Files\Freescale\Freescale MQX 3.1\tools`)
2. Locate the *plugins* directory in the CodeWarrior installation folder (by default `C:\Program Files\Freescale\CodeWarrior for ColdFire V7.1`)
3. Copy the `CFrtos_MQX.dll` file from the MQX *tools* folder to *<CodeWarrior>*`\bin\Plugins\Debugger\rtos`
4. Start or re-start the CodeWarrior Studio.
5. In the CodeWarrior environment, you should be now able to enable MQX TAD by selecting *"MQX"* as *"Target OS"* in the *"CF Debugger Settings"* panel of project settings. All example applications coming with Freescale MQX™ RTOS are already configured so.



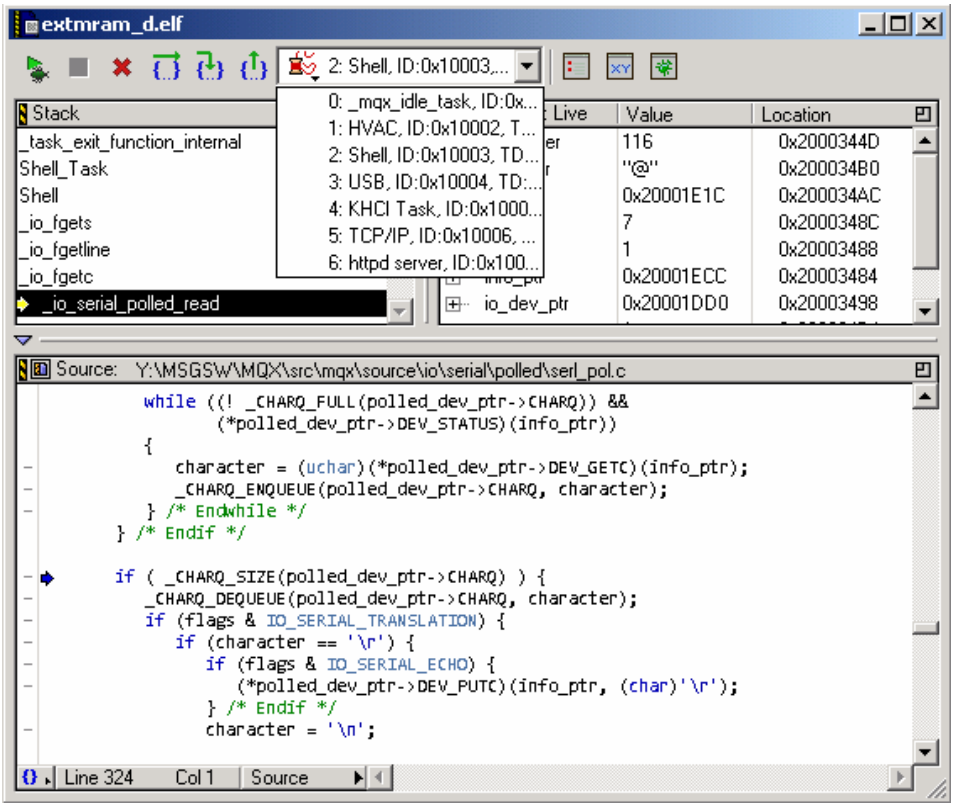
3.11.2 CodeWarrior TAD Features

The TAD plug-in enables more advanced and user-friendly debugging of MQX-based applications.

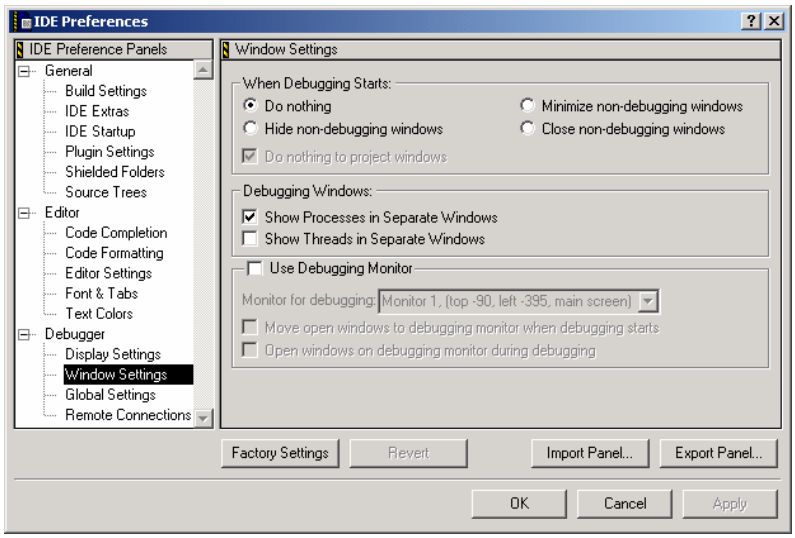
- Using the MQX or RTCS menu in CodeWarrior IDE, several TAD “screens” may be opened during the debugging session. The most helpful and frequently used screens are shown in the picture below:
    - *Task Summary* – overview about all tasks created in the MQX application.
    - *Stack Usage Summary* – displays information about interrupt and task stacks. Typically, stack overflow is a root cause of vast majority of problems in MQX user applications.
    - *Memory Block Summary* – displays address, size and type information about each memory block allocated in the default memory pool by the MQX system or applications. Additional memory pools (if used) may be displayed using “Memory Pools” screen.


MQX
Kernel Data
Check for Errors
Task Summary
Ready Queues
Stack Usage
Memory Pools
Memory Blocks
Memory Extension Blocks
Lightweight Memory Pool
Partition Summary
Message Queues
Message Pools
Lightweight Message Queues
Lightweight Events
Lightweight Semaphores
Events
Mutexes
Semaphores
Task Queues
Initialization
Interrupt Summary
IO Devices
Lightweight Timer Queues
Kernel Log
Log Summary
Names
Check for new MQX version...
About TAD Plug-in...
Close All TAD Windows
Copy Active To Clipboard
RTCS
RTCS Config
Socket Summary
Socket Config
TCP Config
UDP Config
IP Config
NAT Config
TCP Stats
UDP Stats
IP Stats
ICMP Stats
IGMP Stats
ARP Stats
IP-IF Stats
NAT Stats
Ethernet Stats
Task Summary
Task Name Task ID TD Priority State Task Error Code
_mqx_idle_task 0x10001 0x2000210c 13 Ready OK
HVAC 0x10002 0x20002338 9 Active OK
Shell 0x10003 0x200029ac 12 Ready OK
USB 0x10004 0x200035fc 8 Time delay blocked OK
KHCI Task 0x10005 0x20004238 8 LW Msg Rx Blocked, timeout OK
TCP/IP 0x10006 0x20005200 6 Rx Msg Blocked OK
httpd server 0x10007 0x2000a150 8 Msg Send Blocked OK
Stack Usage Summary
Task Stack Base Stack Limit Stack Used % Used Overflow?
_mqx_idle_task 0x20002314 0x20002204 0x20002280 54 % No
HVAC 0x20002988 0x20002410 0x20002640 60 % No
Shell 0x200035d8 0x20002a84 0x20003340 22 % No
USB 0x20003f6c 0x200036d4 0x20003c8c 33 % No
KHCI Task 0x20004970 0x20004330 0x20004678 47 % No
TCP/IP 0x20005eb0 0x200052f8 0x20005ba4 26 % No
httpd server 0x2000aec0 0x2000a248 0x2000aaa8 32 % No
Interrupt 0x200016b4 0x200012b4 0x200015c0 23 % No
Memory Block Summary
Start: 0x20001294
End: 0x2000ffcc
Size: 0x0000ed5c (59.0K)
Highest: 0x2000b5ab (40.0K, 68%)
Address Size Hex Size Dec. Owner Type
0x200012b4 0x404 1028 System Interrupt Stack
0x200016d8 0x108 264 System System Task
0x20001800 0xe0 224 System Ready Qs
0x20001900 0x8 8 System PSP support struct
0x20001928 0x438 1080 System Interrupt Table
0x20001d80 0x30 48 System I/O: Serial polled device struct
0x20001dd0 0x2c 44 System I/O Device
0x20001e1c 0x20 32 System File
0x20001e5c 0x50 80 System I/O: Serial charq
0x20001ecc 0x5c 92 System I/O: Serial info struct
0x20001f48 0x2c 44 System I/O Device
0x20001f94 0x50 80 System
0x20002004 0x2c 44 System I/O Device
0x20002050 0x50 80 System I/O: PCflash
0x200020c0 0x2c 44 System I/O Device
0x2000210c 0x20c 524 0x10001 Task + Stack
0x20002338 0x654 1620 0x10002 Task + Stack
0x200029ac 0xc30 3120 0x10003 Task + Stack

- TAD also provides native debugger support for multi-tasking MQX environment. When an application is stopped at breakpoint or by pressing the (red-square) “Break” button, the name of active task is displayed in the drop-down list in the debugger window. You also have a chance to use the drop-down list to see current execution point of any other task.



Note: This CodeWarrior feature needs to be enabled in general Debugger settings first. Go to the *Edit / Preferences* menu in the CodeWarrior main window and select *“Debugger / Window Settings”* panel. In the *“Debugging Windows”* section, enable the *“Show Processes in Separate Windows”* option and disable the *“Show Threads in Separate Windows”* option.



4 What is Missing?

The MQX development teams in Freescale are working of continuous improvement of the MQX code, bringing new features as well as enabling and validating legacy MQX features with the latest supported devices.

This section provides a list of legacy MQX features which were removed and are not available in the current release of Freescale MQX™ RTOS. Features may either be removed temporarily, until the code is re-tested and validated with the latest supported devices; or permanently when the feature is considered obsolete.

4.1 In MQX PSP

4.1.1 MMU and Virtual Memory Handling

Memory Management Unit and Virtual Memory is not available on devices currently supported by Freescale MQX™ RTOS.

4.1.2 IPC – Communicating Between Processors

The IPC component which enables inter-processor communication, message exchange and synchronization has not yet been validated with devices currently supported by Freescale MQX release.

4.2 In RTCS TCP/IP Stack

4.2.1 RTCS Example Applications

The legacy RTCS example applications as summarized below have been removed from the Freescale MQX™ RTOS distribution and were replaced by other applications.

- **apps** – demonstrating the FTP, Telnet, TFTP and SNMP server services. The FTP and Telnet services are demonstrated with “HVAC” demo applications in the top-level `demo` directory. Standalone FTP server is also demonstrated in the `mfs/examples/mfs_usb` example application, and by the shell `ftp` and `ftpd` commands.
- **boot** – demonstrating the TFTP booting client application was removed, an equivalent is available through the shell `bootp` command.
- **dhcpclnt** – demonstrating the DHCP client features has been replaced by the “ipconfig” shell command implementation (used in HVAC demo). This command uses the new IPCFG feature in RTCS to simplify various kinds of address binding, and by the shell `dhcp` commands..
- **dhcpsrv** – demonstrating DHCP server features has been temporarily removed.
- **lcpecho** – exercising the PPP protocol has been removed.
- **qotdclnt** and **qotdsrv** – showing simple socket communication has been temporarily removed.
- **rtcsdemo** – used as an example of MQX/RTCS application generated by “Application Generator” tool has been removed as this PC Host tool is not available in the current release.

4.3 In USB Support

4.3.1 OTG Support

The USB OTG Drivers are not included in the current release of Freescale MQX™ RTOS.

4.3.2 CDC Class Support

The standard CDC communications class is not supported by either host or device stacks. The device-side implementation is currently under development.

4.4 Peripheral Drivers

Not all peripheral modules available on-chip are supported by an MQX I/O driver.

4.4.1 Cryptographic Acceleration Unit

No legacy driver is available in MQX. This module is not suitable for general purpose device driver. Module features will be used internally by future kernel versions.

4.4.2 Random Number Generator

No legacy driver is available in MQX.

A simple generic I/O driver for this module will be used as an example in Application Note document, demonstrating how to create device driver step-by-step. This driver will be part of future MQX releases.

4.4.3 Flash Module

Driver for writing to external flash module was available in the original MQX RTOS. This driver was removed from the Freescale MQX™ RTOS release until the support for internal processor Flash memory will be implemented.

4.4.4 Various Timer Modules

No legacy driver is available in MQX. This driver is a subject of future development.

4.4.5 I2C and QSPI Communication (Slave Mode)

The I2C and SPI drivers now support Master Mode only. PCB-based drivers were available in the older MQX versions and supported also slave modes. Porting these drivers is a subject of future development.

4.4.6 PWM Module

No legacy driver is available in MQX. This driver is a subject of future development.

4.5 PC Host Tools

PC Host tools known from previous MQX releases (Design tool, EDS client tool, Performance tool) are not included in the current release of Freescale MQX™ RTOS.

5 Known Issues

5.1 Performance of Code Running in MRAM

As described above, the “Ext. MRAM Debug” build-target provides the most convenient way how to download and run code on the M52259EVB board. One limitation of this solution is that the code runtime performance is approximately 8x degraded comparing to Flash-based execution. MRAM is an external memory device connected to the ColdFire core by 8-bit data bus and having one wait-state generated for each access. In order to fetch one 32-bit value (instruction) from the MRAM memory, four accesses need to occur – each inserting one wait-state clock.

This behavior is normal by design.

5.2 Network Communication Performance and Delayed ACKs

Significant differences in performance of host to target TCP communication can be observed, depending on which client operating system is used. These differences are a result of the difference in implementation of the TCP delayed ACK algorithm.

Linux is more aggressive in acknowledging packets, and temporarily disables the delayed ACK algorithm on connection startup. As a result, there is no performance issue observed with Linux-based browsers (Mozilla Firefox tested).

Microsoft Windows based operating systems take a common approach to implementing the delayed ACK algorithm. As a result, web server performance can be significantly worse when it is accessed from Microsoft Windows-based browsers (Internet Explorer and Mozilla Firefox tested). This issue and a workaround are described at http://support.microsoft.com/kb/328890.

This issue is not directly related to the RTCS stack, however limited RAM resources on MCF52259 imply lower buffer and data window sizes, making it more difficult to work around the issue in RTCS software.

5.3 “HVAC” Demo Applications

In the maximal configuration, the HVAC and WEB_HVAC demo applications are supposed to demonstrate simultaneous operation of several key MQX features:

- MQX RTOS real-world multi-tasking application
- MQX BSP and I/O drivers (Ethernet, UART, GPIO)
- RTCS TCP/IP network stack with FTP, Telnet or HTTP server
- USB Host driver with Mass Storage Class Driver
- MFS File System used to access the USB stick directories and files

Total RAM requirements for all of the features are very close to 64k, which is the amount of memory available on MCF52259 device. The application performance may be significantly affected if more RAM needs to be allocated at run-time. Also memory fragmentation may occur after some time , which may cause problems in network or USB communication.

You can use the debugger and the TAD “memory blocks” screen to see the status of the memory pool and amount of remaining free memory.

What may cause higher RAM allocation or memory fragmentation?

- Using USB memory sticks with block size larger than standard 512 bytes. This makes the MFS component to allocate more memory for sector operations.
- Changing the kernel code or configuration. For example, increasing the number of Ethernet buffers or making the TCP buffers larger helps to increase other applications' performance, but makes the HVAC and WEB_HVAC demos unusable. Also UDP protocol and some more advanced features (like DNS) should be disabled to save RAM.
- Attaching/detaching USB device while servicing the TCP connections may increase the memory fragmentation.

Typical failures of the HVAC demos caused by lack of RAM resources include:

- MFS fails to initialize when USB memory stick is attached.
- Web browser may fail to download pictures or other files (style-sheets, script files) referenced in the web page.

Workaround: With M52259EVB board, the HVAC demo can be configured to use part of the MRAM memory as additional RAM. The RTCS and MFS memory "pools" may be allocated in this area by setting DEMOCFG_USE_POOLS macro to 1 in the `hvac.h` header file. With memory pools enabled, the RTCS and WEB_HVAC demo may be configured with UDP/DHCP support.

5.4 Ethernet Link-status Detection

As mentioned above, the IPCFG feature in the RTCS is able to monitor Ethernet link status and re-bind the IP address in static or DHCP mode. In order to be able to read the link status value from the Ethernet PHY device, the MII/MDIO interface needs to be set up properly on each board:

- **M52259EVB Board** – Use jumpers J9 and J10 to route FEC_MDIO and FEC_MDC signals from Ethernet PHY device to MCF52259 processor.
- **M52259DEMO** – The FEC_MDC pin is shared with GPIO signal used to sense the SW1 button state. The Ethernet link status monitoring is not functional in demos which use SW1 button (all HVAC demos).

If MDIO signals are not configured properly, the ethernet link status may be evaluated always "on". This behavior is normal by design.

5.5 Default Kernel Configuration

The default kernel configuration for both M52259EVB and DEMO boards is optimized for HVAC and WEB_HVAC demos. To meet the RAM constraints, some MQX and RTCS features are disabled by default:

- Initialization of TTYB and ITTYB serial device drivers is disabled in the BSP (used in mqx/examples/io).
- Initialization of I2C and QSPI device drivers is disabled in the BSP (used in mqx/examples/i2c and mqx/examples/qspi)
- Initialization of real-time clock (RTC) device driver is disabled in the BSP (used in mqx/examples/rtc).
- Initialization of PCFLASH device driver is disabled in the BSP (used in mfs/examples/mfs_cfcard_usb)
- The ethernet device driver is configured to have only 4 receive and 2 transmit buffers (3 receive buffers on MCF5223x).

5.6 USB Host HUB Examples

The new HUB class support was enabled in HID example applications. The applications run correctly with the USB device attached both directly, or through the hub. However, the example code is still ready to handle a single device only. Combined Mouse+Keyboard demo is able to handle one mouse and one keyboard simultaneously. Using multiple devices of the same kind attached through the hub is not supported.

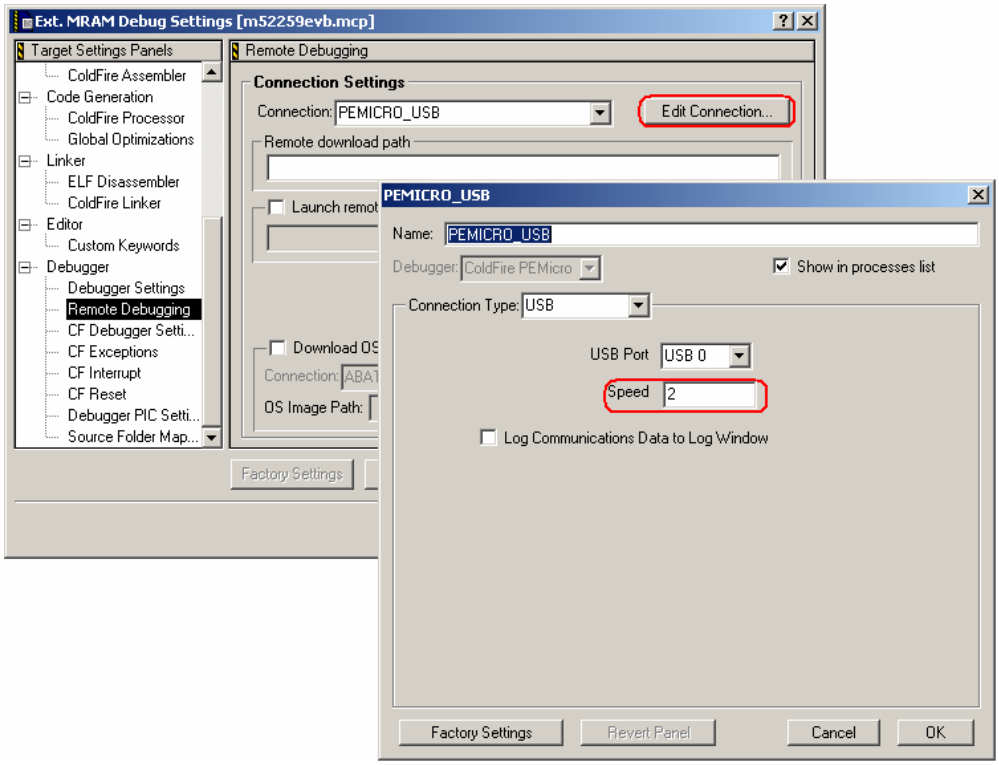
5.7 USB Host HUB Functionality with MCF5222x

The USB HUB functionality issue reported in the MQX release 3.0 (MCF52259) was solved. However the HUB interfacing is still problematic with the older MCF52223. The issue is being investigated and will be addressed by chip errata document.

5.8 BDM Connection Failures with MCF5223x

A problem of BDM communication loss was occasionally observed when debugging M52235EVB applications. This issue is not related to MQX RTOS.

Workaround: This is may be solved by decreasing the BDM communication speed by factor 2 or higher in the "*Remote Debugging*" settings panel for PEMICRO_USB connection. Press the "*Edit Connection*" button and specify the *Speed* factor value "2".



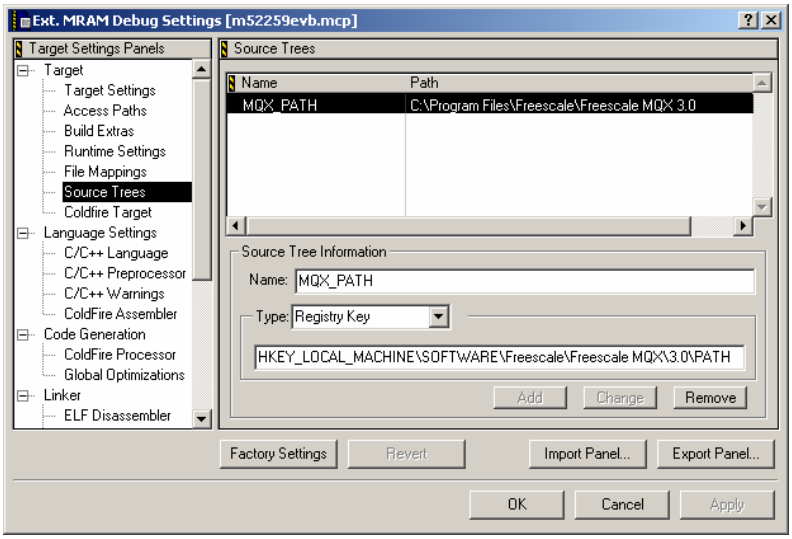
5.9 MQX Stationery / CodeWarrior FlashProgrammer Issue

An issue was identified in the CodeWarrior FlashProgrammer tool. The FlashProgrammer crashes during target board initialization when project uses a "named source tree" in debugger file path. This

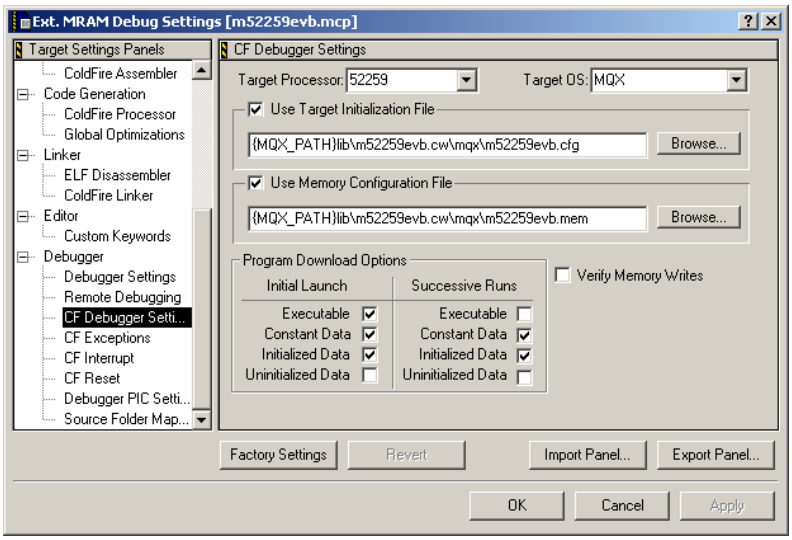
is exactly the case of applications created from MQX stationery. MQX stationery projects use MQX_PATH variable which contains the root path to the MQX installation on the host computer.

This issue was identified after the 3.0 release and remains in the current release. A workaround is provided below.

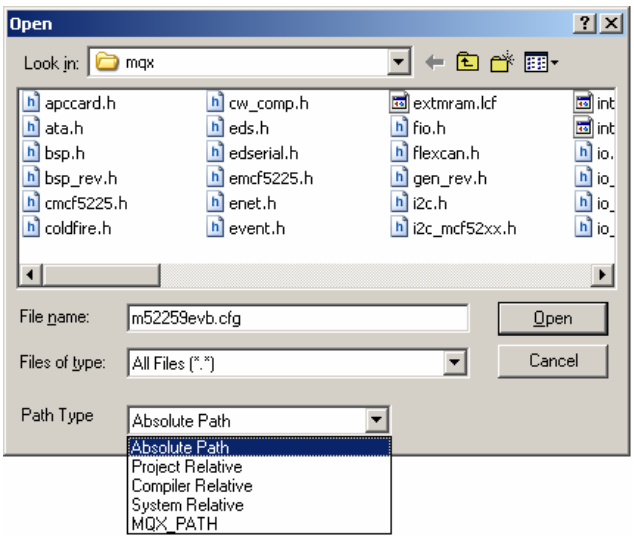
The following picture shows how MQX_PATH variable is defined in the application project created using the MQX CodeWarrior stationery. This variable is then used in several other parameters in the project settings (e.g. see the *Access Paths* panel).



The picture below show how the MQX_PATH variable is used in the *CF Debugger Settings* panel. It substitutes the MQX root installation directory in path to debugger initialization file and memory description file valid for selected BSP. This substitution is not handled correctly by the FlashProgrammer tools.



Workaround: Before using FlashProgrammer tool, the "*Target Initialization File*" path must be changed in a way it does not use the MQX_PATH variable. Press the "*Browse*" button next to the "*Target Initialization File*" edit box and change *Path Type* to "*Absolute Path*":


Open
Look in: mqx
apccard.h
ata.h
bsp.h
bsp_rev.h
cmcf5225.h
coldfire.h
cw_comp.h
eds.h
edserial.h
emcf5225.h
enet.h
event.h
extmram.lcf
fio.h
flexcan.h
gen_rev.h
i2c.h
i2c_mcf52xx.h
File name: m52259evb.cfg
Open
Files of type: All Files (*.*)
Cancel
Path Type
Absolute Path
Absolute Path
Project Relative
Compiler Relative
System Relative
MQX_PATH

6 Change Log

Version 3.0.0 (December 5th 2008)

- This is initial release supporting the MCF52259 processor, M52259EVB and M52259DEMO boards.

Version 3.0.1 (January 22nd 2009)

- Small enhancements throughout the whole code base of MQX kernel and other components
    - Memory block “type” information added to all system memory allocations. TAD is now able to give detailed information about each memory block allocated by kernel or system component.
    - Dedicated memory allocation routines in RTCS, MFS and USB simplify the memory pool usage.
- IPCFG ethernet link monitoring features and automatic IP address binding functionality was added to RTCS. The “ipconfig” shell command replaces the old “bind”, “ifbind” and “dhcp” shell commands.
- The HTTP server in RTCS was re-written to enable multiple sessions to be served by a single task. This feature brings more reliable HTTP server while maintaining low memory requirements.
- USB Host HUB class was added.
- MQX User Guide and several other documents are included in the setup package.

Version 3.1.0 (April 3rd 2009)

- USB Device low-level driver has been implemented for MCF522xx family and example applications were created for M52259EVB and M52223EVB.
- The “usb” subdirectory of the /lib output folder was split to “host” and “device” parts.
- Added PSP, BSPs and other support files for MCF52223 and MCF52235 evaluation boards:
    - USB Host and Device libraries were ported to M52223EVB
    - RTCS library was ported to M52235EVB
    - MFS and Shell libraries ported to both new boards
- USB Host HUB issue resolved with MCF52259 (excessive number of errors observed when USB devices were accessed through USB HUB). The issue was solved by implementation of SOF frame scheduler in the USB Host low-level driver. An issue with a similar impact remains on MCF5222x implementation (silicon problem, no workaround known).
- Error codes naming and numbering convention has been made more consistent across the MQX, RTCS and other libraries. Old error code names remain implemented for backward compatibility (numeric values changed).
- Kernel Data and other internal structures were optimized for size. Parts of the structures were made conditionally compiled based on the user configuration.
- TAD (CodeWarrior debugger plugin) was updated to handle renaming in internal structures.

- Memory type information was added also to lightweight memory structures. TAD is now able to display the type information for memory blocks allocated using lightweight memory allocator.
- Lightweight memory allocator has been made the default option for all BSPs.
- The `/config/<board>/user_config.cw` files were eliminated. Such files have contained a subset of configuration options for assembler-coded files in the MQX kernel. All kernel assembler-coded files are now preprocessed using C preprocessor and make use of standard C header macros. The C-language-specific content of the header files is conditionally excluded during assembler compilation by using `__ASM__` macro.
- The assembler `vectors.cw` file in each BSP was re-coded to C syntax. A new user configuration macro (`MQX_ROM_VECTORS`) may be set non-zero to avoid vector table being copied to RAM.
- The BSP, PSP, RTCS and other library build projects were changed to be easier to understand (virtual folders inside each CodeWarrior project was updated).
- SNMPv2 code was changed to enable ROM-based MIB structures. Example application is provided demonstrating custom MIB nodes and user traps.
- Several source code files were renamed to better reflect the content, especially in the I/O driver directories.
- Added “root directory” concept to the FTP server, “rename” command added.
- SPI I/O driver added (Master mode only). Example application is available.
- Initial ADC I/O driver was added. This driver (including its API) is subject to improvement in future releases. ADC device driver usage is demonstrated in ADC example application.
- RTC API updated, example application added.
- The new IPCFG API updated and documented
- The new HTTP server API documented
- MQX I/O Driver User Guide added